Deciding When and How to Learn

Jonathan Gratch[a], Gerald DeJong[a], and Steve A. Chien[b]

[a]Beckman Institute
University of Illinois
405 N. Mathews Av., Urbana, IL 61801
{gratch, dejong} @cs.uiuc.edu

[b]Jet Propulsion Laboratory,
California Institute of Technology,
4800 Oak Grove Drive, M/S 525-3660
Pasadena, CA 91109-8099

Abstract

Learning is an important aspect of intelligent behavior. Unfortunately, learning rarely comes for free. Techniques developed by machine learning can improve the abilities of an agent but they often entail considerable computational expense, Furthermore, there is an inherent tradeoff between the power and efficiency of learning, More powerful learning approaches require greater computational resources. This poses a dilemma to a learning agent that must act in the world under a variety of resource constraints. This paper investigates the issues involved in constructing a rational learning agent. Drawing on work in decision-theory we describe a framework for a rational agent that embodies *learning actions* that can modify its own behavior. The agent must posses deliberative capabilities to assess the relative merits of these actions in the larger context of its overall behavior and resource constraints. We then sketch several algorithms that have been developed within this framework.

1. INTRODUCTION

A principal component of intelligent behavior is an ability to learn. In this article we investigate how machine learning approaches relate to the design of a *rational learning agent.* This is an entity that must act in the world in the service of goals and includes among its actions the ability to change its behavior by learning about the world. There is a long tradition of studying goal directed action under variable resource constraints in economics [Good71] and the decision sciences [Howard70], and this forms the foundations for artificial intelligence approaches to resource-bounded computational agents [Doyle90, Horvitz89, Russell89]. However, these AI approaches have not addressed many of the issues involved in constructing a *learning* agent. When learning has been considered, it has involved simple, unobtrusive procedures like passively gathering statistics [Wefald89] or off-line analysis, where the cost of learning is safely discounted [Horvitz89]. Many machine learning approaches require extensive resources and may force an agent to redirect its behavior temporarily away from satisfying its goals and towards exploratory behavior. Thus, more recent work has begun to consider the problem of developing techniques that can make a reasoned compromise between learning and acting, and more generally, to consider the relationship between learning and the overall goals and resource constraints of a rational agent [desJardins92, Gratch93b, Provost92].

Deeiding when and how to learn is made difficult by the realization that there is no "universal" learning approach. Machine learning approaches strike some balance bet ween two conflicting objectives. First is the goal to create powerful learning approaches to facilitate large performance improvements. Second is the need to restrict approaches for pragmatic considerations such as efficiency. Learning algorithms implement some *bias* which embodies a particular compromise between these objectives. Unformately, in what is called the ***static bias problem,*** there is no single bias that applies equally well across all domains, tasks, and situations [Provost92]. As a consequence there is a bewildering array of learning approaches with differing tradeoffs between power and pragmatic considerations.

This tradeoff poses a dilemma to any agent that must learn in complex environments under a variety of resource constraints. Practical learning situations place varying demands on learning techniques. One case may allow several CPU months to devote to learning; another requires an immediate answer. For one case training examples are cheap; in another, expensive procedures are required to obtain data. An agent wishing to use a machine learning approach must decide which of many competing approaches to apply, if any, given its current circumstances. This evaluation is complicated because techniques do not provide feedback on how a given tradeoff plays out in specific domains. This same issue arises

1. Gerald Tesauro noted that his reinforcement learning approach to learning strategies in backgammon required a month of learning time on an IBM RS/6000 [Tesauro92].

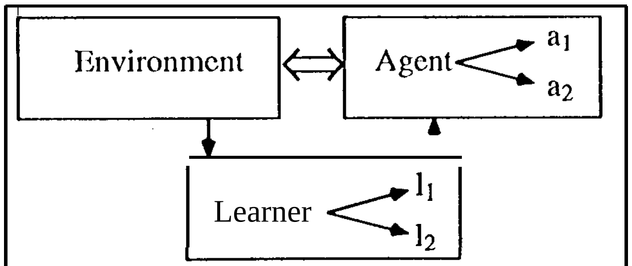

Figure 1a: Standard decision-theoretic view. Learning is considered external to the agent.

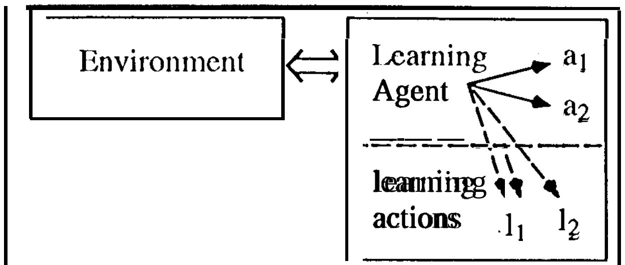

Figure 1 b: A learning agent. Learning is under the direct control of the agent.

in a number of different learning problems. In statistical mod·cling the issue appears in how to choose a mode] that balances the tradeoff between the fit to the data and the number of examples required to reach a given level of predictive accuracy [AI&STAT93]. In neural networks the issue arises in the choice of network architecture (e.g. the number of hidden units). The tradeoff is at the core of the field of decision analysis where formalisms have been developed to determine the value of experimentation under a variety of conditions [Howard70].

This article outlines a decision-theoretic formalism with which to view the problem of flexibly biasing learning behavior. A learning algorithm is assumed to have some choice in how it can proceed and behavior in a given situation is determined by a rational cost-benefit analysis of these alternative "learning actions." A chief complication in this approach is assessing the consequences of different actions in the context of the current situation. Often these consequences cannot be inferred simply on the basis of prior information. The solution we outline in this article uses the concept of *adaptive bias.* In this approach a learning agent uses intermediate information acquired in earlier stages of learning to guide later behavior. We instantiate these notions in the area of speed-up learning through the implementation of a "speed-up learning agent." The approach automatically assesses the relative benefits and costs of learning to improve a problem solver and flexibly adjusts its tradeoff depending on the resource constraints placed upon it.

2. RATIONALITY IN LEARNING

Decision theory provides a forma] framework for rational decision making under uncertainty (see [Berger80]). Recently, there has been a trend to provide a decision-theoretic interpretation for learning algorithms [Brand91, Gratch92, Greiner92, Subramanian92]. The "goal" of a learning system is to improve the capabilities of some computational agent (e.g. a planner or classifier). The decision-theoretic perspective views a learning system as a decision maker, The learning system must decide how to improve the computational agent with respect to a fixed, but possibly unknown, distribution of tasks. The learning system represents a space of possible *transformations* to the agent: a space of alternative control strategies for a speed-up learning system or a space of possible concepts for a classification learning system. Each of the transformations imparts some behavior to the computational agent on the tasks it performs. The quality of a behavior is measured by a *utility function* that indicates how well the behavior performs on a given task. The *expected utility* of a behavior is the average utility over the fixed distribution of tasks. Ideally, a learning system should choose the transformation that maximizes the expected utility of the agent, To make this decision, the learning system may require many examples from the environment to estimate empirically the expected utility of different transformations.

1.1 Learning an Agent vs. a Learning Agent

Figure 1a illustrates this view of learning systems. There is a computational agent that must interact with an environment. The agent exhibits some behavior that is characterized by the expected utility of its actions with respect to the particular environment. For example the agent may bean automated planning system, the environment a distribution of planning problems, and the behavior characterized by the average time required to produce a plan. A learning algorithm decides, from a space of transformations, how best to change the behavior of the agent, This view clarifies the purpose of learning: the learning algorithm should adopt a transformation that maximizes the expected utility of the agent's behavior, This view does not, however, say anything about how to balance pragmatic considerations. The utility of the agent is defined without regard to the learning cost required to achieve that level of performance. Pragmatic issues are addressed by an external entity through the choice of a learning algorithm.

Figure lb illustrates the view we explore in this article. Again there is a computational agent that must interact with an environment. Again the agent exhibits behavior that is characterized by the expected utility of its actions. The difference is that the decision of what and how to learn is under control of the agent, and consequently, pragmatic considerations must also be assessed by the agent, and cannot be handed over to some external entity. In addition to normal actions, it can perform learning actions that change its behavior towards the environment. A learning action might involve running a machine learning program as a black box, or it might involve direct control over low-level learning operations – thereby giving the agent more flexibility over the learning process. The expense of learning may well negatively impact expcctcd utility. Therefore, the learning agent must be *reflective* [Horvitz89]. This means it must possess the means to reason about both the benefits and costs of possible learning actions. As an example, such a learning agent could be a automated planning system with learning capabilities. After solving each plan this system has the opportunit y to expend resources analyzing its behavior for possible improvements. The decision of what and how to learn must involve consideration of how the improvement and the computational expense of learning impact the average utility exhibited by the agent,

Note that the sense of utility differs in the two views. In the first case, utility is defined independent of learning cost. In the second case, utility must explicitly encode the relationship between the potential benefits and costs of learning actions. For example, the average time to produce a plan may be a reasonable way to characterize the behavior of a non-learning agent but it does not suffice to characterize the behavior of a learning agent. There is never infinite time to amortize the cost of learning. An agent operates under finite resources and learning actions are only reasonable if the improvement they provide outweighs the investment in learning cost, given the resource constraints of the present situation. These two senses of utility are discussed in the rational decision making literature. Following the terminology of Good we refer to a utility function that does not reflect the cost of learning as Type I utility function [Good71]. A utility function that dots reflect learning cost is a Type 11 utility function. The Type I vs. Type 11 distinction is also referred to as substantive vs. procedural utility [Simon76] or object-level vs. comprehensive utility [Horvitz89]. Wc arc interested in agents that assess Type 11 utility.

1.2 Examples of Type II Utility

Type I functions can be used in realistic learning situations only after some human expert has assessed the current resource constraints, and chosen a learning approach to reflect the appropriate tradeoff. The decision-theoretic view is that the decision making of the human expert can be modeled by a Type 11 utility function. Thus, to develop a learning agent wc must capture the relationship between benefit and cost that wc believe the human expert is using. This, of course, will vary with different situations, We discuss a few likely situations and show how these can be modeled with Type 11 functions.

Speed-up learning is generally cast as reducing the average amount of resources needed by an agent to solve problems. The process of learning also consumes resources that might otherwise be spent on problems solving. A Type 11 function must relate the expense of this learning investment and its potential return. If wc are restricted to a finite resource limit, a natural goal is to evaluate the behavior of the learning agent by how many problems it can solve within the resource limit. A learning action is only worthwhile if leads to an increase in this number. In this case the expcctcd Type II utility corresponds to the expected number of problems it can solve given the resource limit. If we are given a finite number of problems to solve, an alternative relationship might be to solve them in as few resources as possible. In this case expected Type 11 would be the expected resources (including learning resources) required to solve the fixed number of problems. Learning actions should be chosen only if they will reduce this value.

Another situation arises in the case of learning a classifier where there is great expense in obtaining training examples. The presumed benefit of a learned classifier is that it enables an agent to make classifications relevant to achieving its goals. Obviously, the more accurate the classifier, the greater the benefit. It is also generally the case that a more accurate classifier will result by increasing the richness of the hypothesis space – increasing the number of attributes in a decision tree, increasing the number of hidden units in a neural network, or increasing the number of parameters in a statistical modeling procedure. While this improves the potential benefit, adding flexibility will increase the number of examples required to achieve a stable classifier. A Type 11 utility function should relate the expected accuracy to the cost of obtaining training examples. A learning system should then adjust the flexibility (attributes, hidden units, etc.) in such a way to maximize this Type II function.

1.3 Adaptive Bias

In our model an agent must be able to assess the costs and benefits of alternative learning actions as they relate to the current learning task. Occasionally it maybe possible to provide the learning agent with enough information in advance to make such determinations (e.g. ***uniformities*** are one source of statistical knowledge that allows an agent to infer the benefits of using a given bias [desJardins92]). Unfortunately, such information is frequently unavailable. We introduce the notion of adaptive bias to enable flexible control of behavior in the presence of limited information on the cost and benefit of learning actions. With adaptive bias a learning agent estimates the factors influences cost and benefit as the learning process unfolds. Information gained in the early stages of learning is used to adapt subsequent learning behavior to the characteristics of the given learning context.

3. RATIONAL SPEED-UP LEARNING

The notion of a rational learning agent outlined in Section NO TAG is quite general. Many interesting, and more manageable learning problems arise from imposing assumptions on the general framework. We consider one specialization of these ideas in some detail in this section, while the following sect ion relates several alternative approaches.

One of our primary interests is in area of speed-unlearning and therefore we present an approach related to this class of learning problems. We adopt the following five assumptions to the general model. (1) An agent is viewed as a problem solver that must solve a sequence of tasks provided by the environment according to a fixed distribution. (2) The agent possesses learning actions that change the average resources required to solve problems. (3) Learning actions must precede any other actions. This follows the typical characterization of the speed-up learning paradigm: there is an initial learning phase where the learning system consumes training examples and produces an improved problem solver; this is followed by a utilization phase where the problem solver is used repeatedly to solve problems. Once stopped, learning may not be returned to. (4) The agent learns by choosing a sequence of learning actions and each action must (probabilistically) improve expected utility. As a consequence, the agent cannot choose to make its behavior worse in the hope that it may produce larger subsequent improvements. (5) Learning actions have access to an unbounded number of training examples. For any given run of the system, this number will be finite because the agent must act under limited resources. This is reasonable for speed-up learning techniques are unsupervised learning techniques so the examples do not need to declassified. Together, these assumptions define a sub-class of rational learning agents we call *one-stage speed-up learning agents.*

Given this class of learning agents, their behavior is determined by the choice of a Type II utility function, We have developed algorithms for two alternative characterizations of Type II utility that result from different views on how to relate benefit and cost. These characterizations essentially amount to Type 11 utility schemas – they combine some arbitrary Type I utility function and an arbitrary characterization of learning cost into a particular Type II function. Our first approach, RASL, implements a characterization which leads the system to maximize the number of tasks it can solve with a fixed amount of resources. A seeond approach, RIBS, encodes a different characterization that leads the system to achieve a fixed level of benefit with a minimum of resources. Both approaches are conceptually similar in their use of adaptive bias, so we describe RASL in some detail to give a flavor of the techniques,

RASL (for RAtional Speed-up Learner) builds upon our previous work on the COMPOSER learning system, COMPOSER is a general statistical framework for probabilistically identifying transformations that improve Type I utility of a problem solver [Gratch91, Gratch92]. COMPOSER has demonstrated its effectiveness in artificial planning domains [Gratch92] and in a real-world scheduling domain [Gratch93a]. The system was not, however, developed to support rational learning agents. It has a fixed tradeoff for balancing learned improvements with the cost to attain them and provides no feedback, Using the low-level learning operations of the COMPOSER system, RASL incorporates decision making procedures that provide flexible control of learning in response to varying resource constraints. RASL acts as a learning agent that, given a problem solver and a fixed resource constraint, balances learning and problem solving with the goal of maximizing expected Type 11 utility. In this context of speed-up learning resources are defined in terms of computational time, however the statistical model applies to any numeric measure of resource use (e.g., space, monetary cost).

1.4 Overview

RASL is designed in the service of the following Type 11 goal: Given a problem solver and a known, fixed amount of resources, solve as many problems as possible within the resources, RASL can start solving problems immediately. Alternatively, the agent can devote some resources towards learning in the hope that this expense is more than made up by improvements to the problem solver. Type II utility corresponds to the number of problems that can be solved with the available resources, The agent should only invoke learning if it increases the *expected number of problems* (*ENP*) that can be solved. RASL embodies decision making capabilities to perform this evaluation.

The expected number of problems is governed by the following relationship:

$$ENP = \frac{\text{available resources after learning}}{\text{avg. resources to solve a problem after learning}}$$

The top term reflects the resources available for problems solving. If no learning is performed, this is simply the total available resources. The bottom term reflects the average cost to solve a problem with the learned problem solver. RASL is reflective in that it guides its behavior by explicitly estimating *ENP*. To evaluate this expression, RASL must develop estimators for the resources that will be spent on learning and the performance of the learned problem solver,

Like COMPOSER, RASL adopts a hill-climbing approach to maximizing utility. The system incrementally learns transformations to an initial problem solver. At each step in the search, a transformation generator proposes a set of possible changes to the current problem solver.[2] The system then evaluates these changes with training examples and chooses one transformation that improves expected utility. At each step in the search, RASL has t wo options: (1) stop learning and start solving problems, or (2) investigate new transformations, choosing one with the largest expected increase in *ENP*. In this way,

2. The approach provides a framework that can incorporate varying mechanisms to proposing changes. We have developed instantiation of the framework using PRODIGY/EBL's proposed control rules [Minton88], STATIC proposed control rules [}3tzioni91] and a library of expert proposed scheduling heuristics [Gratch93a].

the behavior of RASL is driven by its own estimate of how learning influences the expected number of problems that may bc solved. The system only continues the hill-climbing search as long as the next step is expected to improve the Type II utility. RASL proceeds through a series of decisions where each action is chosen to produce the greatest increase in *ENP*. However, as it is a hill-climbing system, the *ENP* exhibited by RASL will bc locally, and not necessarily globally, maximal.

1.5 Algorithm Assumptions

RASL relies on several assumptions. The most important assumption relates to its reflective abilities. The cost of learning actions consists of three components: the cost of deliberating on the effectiveness of learning, the cost of proposing transformations, and the cost of evaluating proposed transformaions. When RASLdecides on the utility of a learning action, it should account for all of these potential costs. Our implementation, however, relies on an assumption that the cost of evaluation dominates other costs. This assumption is reasonable in the domains COMPOSER has been applied. The preponderant learning cost lies in the statistical evaluation of transformations, through the large cost of processing each example problem — this involves invoking a problem solver to solve the example problem. The decision making procedures that implement the Type II deliberations involve evaluating relatively simple mathematical functions. This assumption has the consequence that RASL does not account for some overhead in the cost of learning, and under tight resource constraints the system may perform poorly. RASL also embodies the same statistical assumptions adopted by COMPOSER. The statistical routines must estimate the mean of various distribution. While no assumption is made about the form of the distributions, we make a standard assumption that the sample mean of these distributions is normally distributed.

1.6 Estimating *ENP*

The core of RASL's rational analysis is a statistical estimator for the *ENP* that results from possible learning actions. This section describes the derivation of this estimator. At a given decision point, RASL must choose between terminating learning or investigating a set of transformations to the current problem solver. RASL uses COMPOSER'S statistical procedure to identify beneficial transformations. Thus, the cost of learning is based on the cost of this procedure. After reviewing some statistical notation we describe this statistical procedure. We then develop from this statistic a estimator for *ENP*.

Let X bc a random variable, An observation of a random variable can yield one of a set of possible numeric outcomes where the likelihood of each outcome is determined by an associated probability distribution. X_i is the ith observation of X. *EX* denotes the expected value of X, also called the mean of the distribution. $\bar{X}_n$ is the *sample mean* and refers to the average of n observations of X. $\bar{X}_n$ is a good estimator for *EX*. *Variance is* a measure of the dispersion or spread of a distribution. We use the *sample variance*, $S^2{}_n = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2$, as an estimator for the variance of a distribution.

The function $\Phi(x) = \int_{-\infty}^{x}\left(1/\sqrt{2\pi}\right)exp\{-0.5y^2\}dy$ is the cumulative distribution function of the standard normal (also called standard gaussian) distribution. $\Phi(x)$ is the probability that a point drawn randomly from a standard normal distribution will bc less than or equal to x. This function plays a important role in statistical estimation and inference. The Central Limit Theorem shows that the difference between the sample mean and true mean of an arbitrary distribution can be accurately represented by a standard normal distribution, given sufficiently large sample. In practice we can perform accurate statistical inference using a "normal approximation."

1.6,1 *COMPOSER's Statistical Evaluation*

Let T be the set of hypothesized transformations. Let *PE* denote the current problem solver. When RASL processes a training example it determines, for each transforrnation, the change in resource cost that the transforrnation would have provided if it were incorporated into *PE*. This is the difference in cost between solving the problem with and without incorporating the transformation into *PE*. This is denoted $\Delta r_i(t|PE)$ for the ith training example. Training examples are processed incremental y. After each example the system evaluates a statistic called a *stopping rule*. The particular rule we use was proposed by Arthur Nádas and is called the Nádas stopping rule [Nadas69]. This rule determines when enough examples have been processed to state with confidence that a transformation will help or hurt the average problem solve speed of *PE*.

After processing a training example, RASL evaluates the following rule for each transforrnation:

$$n \geq n_0 \quad AND \frac{S_n^2}{\left[\overline{\Delta r_n}(t|PE)\right]^2} \leq \frac{n}{a^2}, \quad s.t.\ \Phi(a) = \frac{\delta}{2|T|} \quad (1)$$

where n_0 is a small finite integer indicating an initial sample size, *n is* the number of examples taken so far, is the transformation's average improvement, is the observed variance in the transformation's improvement, and δ is a confidence parameter. If the expression holds, the transformation will speed-up (slow down) *PE* if is positive (negative) with confidence $1-\delta$. The number of examples taken at the point when Equation 1 holds is called the *stopping* time associated with the transformation and is denoted $ST(t|PE)$.

1.6.2 Estimating Stopping Times

Given that we arc using the slopping rule in Equation 1, the cost of learning a transformation is the stopping time for that transformation, times the average cost to process an example. Thus, one element of an estimate for learning cost is an estimate for stopping times. In Equation 1, the stopping time associated with a transformation is a function of the variance, the square of the mean, and the sample size n. We can estimate these first two parameters using a small initial sample of n_0 examples: and $\overline{\Delta r_n}(t|PE) \approx \overline{\Delta r_{n_0}}(t|PE)$. We can estimate the stopping time by using these estimates, treating the inequality in Equation 1 as an equality, and solving for n. The stopping time cannot be less than the n_0 initial examples so the resulting estimator is:

$$\hat{ST}_{n_0}(t|PE) = \max\left\{ n_0, \left\lceil a^2 \frac{S^2_{n_0}(t|PE)}{\left[\overline{\Delta r_{n_0}}(t|PE)\right]^2} \right\rceil \right\} \quad (2)$$

where is the average improvement in Type I utility that results if transformation t is adopted, is the variance in this random variable, and $\Phi(a) = \delta/(2|T|)$.

1.6.3 Learning Cost

Each transformation can alter the expected utility of a performance element. To accurately evaluate potential changes we must allocate some of the available resources towards learning. Under our statistical formalization of the problem, learning cost is a function of the stopping time for a transformation, and the cost of processing each example problem.

In the general case, the cost of processing the jth problem depends on several factors. It can depend on the particulars of the problem. In can also depend on the currently transformed performance element, PE_i. For example, many learning approaches derive utility statistics by executing (or simulating the execution) of the performance element on each problem. Finally, as potential transformations must be reasoned about, learning cost can depend on the size of the current set of transformations, 1.

Let $\lambda_j(T, PE)$ denote the learning cost associated with the jth problem under the transformation set T and the performance element PE. The total learning cost associated with a transformation, t, is the sum of the per problem learning costs over the number of examples needed to apply the transformation:

1.6.4 ENP Estimator

We can now describe how to compute this estimate for individual transformations based on an initial sample of no exampies. Let R be the available resources. The resources expended learning a transformation can be estimated by multiplying the average cost to process an example by the stopping time associated with that transformation: . The average resource usc of the learned performance element can be estimated by combining estimates of the resource use of the current performance clement and the change in this use provided by the transformation: $\overline{r}_{n_0}(PE) - \overline{\Delta r_{n_0}}(t|PE)$, Combining these estimators yields the following estimator for the expected number of problems that can be solved after learning transformation t:

$$\hat{ENP}_{n_0}(R, t|PE) = \frac{R - \lambda_{n_0}(t, T, PE) \times \hat{ST}_{n_0}(t|PE)}{\overline{r}_{n_0}(PE) - \overline{\Delta r_{n_0}}(t|PE)} \quad (3)$$

Space precludes a description of how this estimator is used but the RASL algorithm is listed in the Appendix, Given the available resources, RASL estimates, based on a small initial sample, if a transformation should bc learned. If not, is uses the remaining resources to solve problems. If learning is expected to help, RASL statistically evaluates competing transformations, choosing the one with the greatest increase in *ENP*. It then determines, based on a small initial sample, if a new transformation should be learned, and so on.

4. EMPIRICAL EVALUATION

RASL'S mathematical framework predicts that the system can appropriately control learning under a variet y of t i me pressures. Given an initial problem solver and a fixed amount of resources, the system should allocate resources between learning and problem solving to solve as many problems as possible. The theory that underlies RASL makes several predictions that we can evaluate empirically, (1) Given an arbitrary level of initial resources, RASL should solve a greater than or equal to number of problems than if we used all available resources on problem solving[3]. As we increase the available resources, RASL should (2) spend more resources learning, (3) acquire problem solvers with lower average solution cost, (4) exhibit a greater increase in the number of problems that can be solved,

Wc empirically test these claims in the real-world domain of spacecraft communication scheduling. The task is to allocate communication requests between earth-orbiting satellites and the three 26-meter antennas at Goldstone, Canberra, and Madrid. These antennas make up part of NASA's Deep Space Network. Scheduling is currently performed by a human expert, but the Jet Propulsion Laboratory (JPL) is developing a heuristic scheduling technique for this task. In a previous article, we describe an application of COMPOSER to improving this scheduler over a distribution of scheduling problems.

3. RASL pays an overhead of taking n_0 initial examples to make its rational deliberations. If learning provides no benefit, this can result in RASL solving less examples than a non-learning agent.

COMPOSER acquired search control heuristics that improved the average time to produce a schedule [Gratch93a]. We apply RASL to this domain to evaluate our claims.

COMPOSER explored a large space of heuristic search strategies for the scheduler using its statistical hill-climbing technique. In the course of the experiments we constructed a large database of statistics on how the scheduler performed with different search strategies over many scheduling problems. We usc this database to reduce the computational cost of the current set of experiments. Normally, when RASL processes a training example it must evoke the scheduler to perform an expensive computation to extract the necessary statistics. For these experiments we draw problems randomly from the database and extract the statistics already acquired from the previous experiments, RASL is "told" that the cost to acquire these statistics is the same as the cost to acquire them in the original COMPOSER experiments.

A learning trial consists of fixing an amount of resources, evoking RASL, and then solving problems with the learned problem solver until resources are exhausted. This is compared with a non-learning approach that simply uses all available resources to solve problems. Learning trials are repeated multiple times to achieve statistical significance. The behavior of RASL under different time pressures is evaluated by varying the amount of available resources. Results are averaged over fifty learning trials to show statistical significance. We measure several aspects of RASL'S behavior to evaluate our claims. The principle component of behavior is the number of problems that are solved. In addition we track the time spent learning, the Type I utility of the learned scheduler, and the number of hill-climbing steps adopted by the algorithm. RASL has two parameters. The statistical error of the stopping rule is controlled by δ, which we set at 5%. RASL requires an initial sample of size n_0 to estimate *ENP*. For these experiments we chose a value of thirty. RASL is compared against a non-learning agent that uses the original expert control strategy to solve problems. The non-learning agent devotes all available resources to problem solving.

The results are summarized in Figure 2. RASL yields an increase in the number of problems that are solved compared to a non-learning agent, and the improvement accelerates as the resources are increased, As the resources are increased, RASL spends more time in the learning phase, taking more steps in the hill-climbing search, and acquiring schedulers with better average problem solving performance.

The improvement in *ENP is* statistically significant. The graph of problems solved shows 95% confidence intervals on the performance of RASL over the learning trials. Intervals arc not show for the non-learning system as the variance was negligible over the fifty learning trials. Looking at Type I utility, the search strategies learned by RASL were considerably worse than the average strategy learned in our COMPOSER experiments [Gratch93a]. In this domain, COMPOSER learned strategies that take about thirty seconds to solve a scheduling problem. This suggests that wc must considerably increase the available resources before the cost to attain this better strategy becomes worthwhile." Wc arc somewhat surprised by the strict linear relationship between the available resources and the learning time. At some point learning time should level off as learning produces diminishing returns. However, as suggested by the difference in the schedulers learned by COMPOSER and RASL, we can greatly increase the available resources before RASL reaches this limit.

It is an open issue how best to set the initial sample parameter, n_0. The size of the initial sample influences the accuracy of the estimate of *ENP*, which in turn influences the behavior of the system. Poor estimates can degrade RASL's decision making capabilities. Making n_0 very large will increase the accuracy of the estimates, but increases the overhead of the technique. It appears that the best setting for n_0 depends on characteristics of the data, such as the variance of the distribution. Further experience on real domains is needed to assess the impact of this sensitivity. There are ways to address the issue in a principled way (e.g. using cross-validation to evaluate different setting) but these would increase the overhead of the technique.

5. RELATED WORK

RASL illustrates just one of many possible ways to reason about the relationship between the cost and benefit of learning actions. Alternatives seise when we consider different definitions for Type 11 utility or when there exists prior knowledge about the learning actions.

In [Chien94] we describe the RIBS algorithm (Rational Interval-Based Selection) for a different relationship between cost and benefit, In this situation the Type II utility is defined in terms of the ratio of Type I utility and cost. We introduce a general statistical approach for choosing, with somefixed level of confidence, the best of a set of *k* hypotheses, where the hypotheses are characterized in terms of some Type I utility function. The approach attempts to identify the best hypothesis by investigating alternatives over variably priced training examples, where there is differing cost in evaluating each hypothesis. This problem arises, for example, in learning heuristics to improve the quality of schedules produced by an automated scheduling system.

This issue of rational learning has frequently been cast as a problem of *bias selection* [desJardins92, Provost92]. A learning algorithm is provided with a set of biases and some metaprocedure must choose one bias that appropriately addresses the pragmatic factors. Provost [Provost92] shows that under very weak prior information about bias space, a technique called *iterative weakening* can achieve a fixed level of benefit with near minimum of cost. The approach requires that biases be ordered in terms of increasing cost and expressiveness.

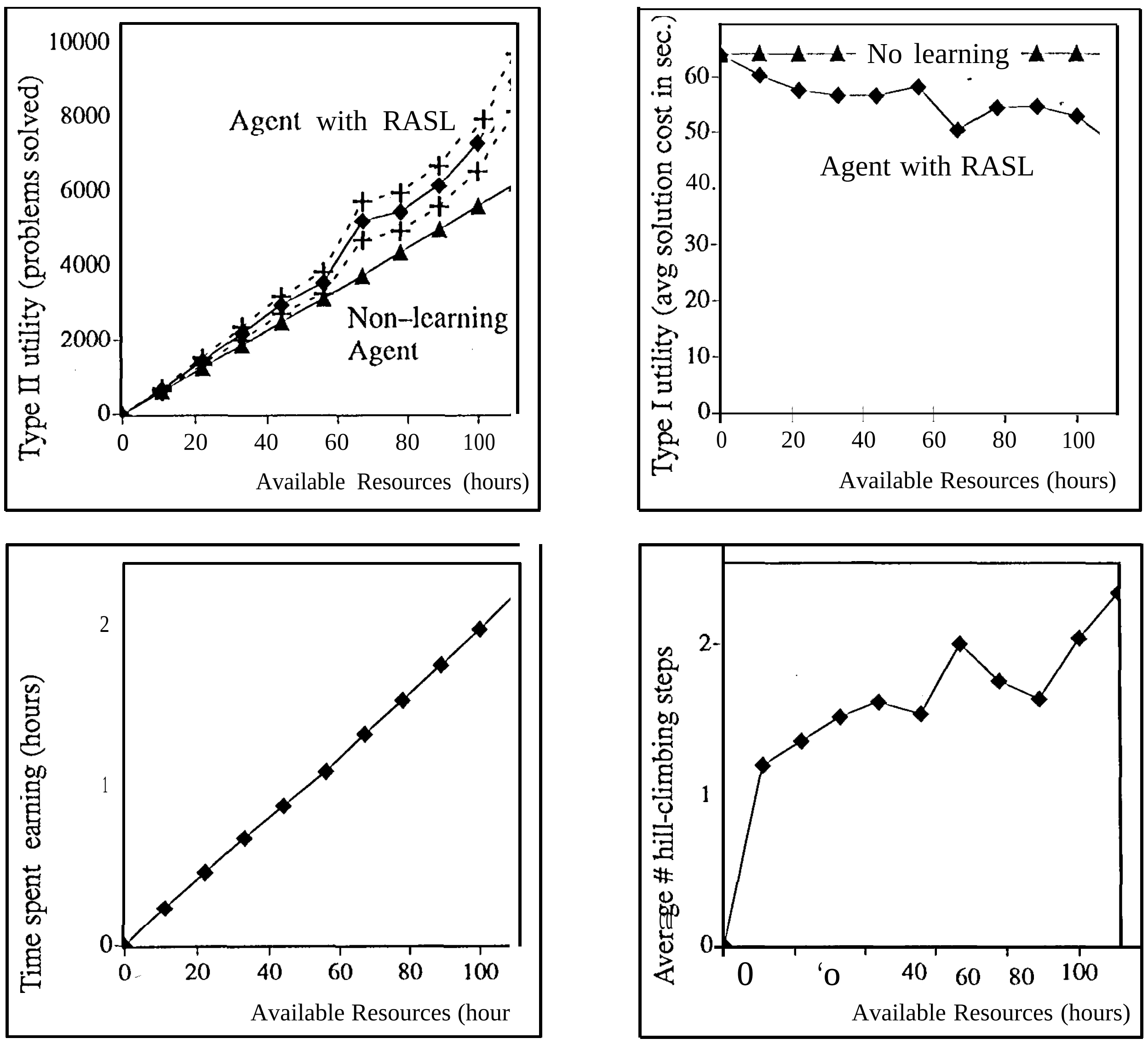

Figure 2: RSL resul Results arc averaged over thirty trials.

desJardins [desJardins92] introduces a class of prior knowledge that can be exploited for bias selection. She introduces the statistical knowledge called uniformities from which one can deduce the predictive accuracy that will result from learning a concept with a given inductive bias. By combining this with a simple model of inductive cost, here PAGODA algorithm can choose a bias appropriate to the given learning situation.

RASL focuses on one type of learning cost that was fairly easy to quantify: the cost of statistical inference. It remains to apply these concepts to other learning approaches. For example, STATIC is a speed-up learning approach that generates knowledge through an extensive analysis of a problem solver's domain theory [Etzioni91]. We might imagine a similar rat ional control for STATIC where the system performs different levels of analysis when faced with different time pressures. The challenge is to develop ways of estimating the cost and benefit of this type of computation. In classification learning, for another example, a rational classification algorithm must develop estimates on the cost and the resulting benefit of improved accuracy that result from growing a decision tree.

This work focused on a one learning tradeoff but there are more issues we can consider. We focused on the computational cost to process training examples. We assume an unbounded supply of training examples to draw upon, however under realistic situations we often have limited data and it can be expensive to obtain more. Our statistical evaluation techniques try to limit both the probability of adopting a transformation with negative utility and the probability of rejecting a transformation with positive utility (the difference between type I and type 11 error). Under some circumstance, one factor may be more important than the other. These issues must be faced by any rational agent that includes learning as one of its courses of action. Unfortunately, learning techniques embody arbitrary and often unarticulated commitments to balancing these characteristics. This research takes just onc of many needed steps towards making learning techniques amenable to rational learning agents,

6. CONCLUSIONS

Traditionally, machine learning techniques have focused on improving the utility of a performance element without much regard to the cost of learning, except, perhaps, requiring that algorithms bc polynomial. While polynomial run time is necessary, it says nothing about how to choose amongst a variety of polynomial techniques when under a variety of time pressures. Yet different polynomial techniques can result in vast practical differences. Most learning techniques are inflexible in the face of different learning situations as they embody a fixed tradeoff between the expressiveness of the technique and its computational efficiency, Even when techniques are flexible, a rational agent is faced with a complex and often unknown relationship between the configurable parameters of a technique and its learning behavior. Wehavepresented an initial foray into rational learning agents. We presented the RASL algorithm, and extension of the COMPOSER system, that dynamically balances the costs and benefits of learning.

Acknowledgements

Portions of this work were performed at the Beckman Institute, University of Illinois under National Science Foundation Grant IRI–92–09394, and at the Jet Propulsion Laboratory, California institute of Technology under contract with the National Aeronautics and Space Administration.

References

[AI&STAT93] AI&STAT, *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, January 1993.

[Berger80] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis,* Springer Verlag, 1980.

[Brand91] M. Brand, "Decision-Theoretic Learning in an Action System, " *Proceedings of the Eighth International Workshop on Machine Learning,* Evanston, IL, June 1991, pp. 283-287.

[Chien94] S. A. Chien, J. M. Gratch and M. C. Burl, "On the Efficient Allocation of Resources for Hypothesis Evaluation in Machine Learning: A Statistical Approach," *submitted to Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence, 1994.*

[desJardins92] M. E. desJardins, "PAGODA: A Model for Autonomous Learning in Probabilistic Domains," Ph.D. Thesis, Computer Science Division, University of California, Berkeley, CA, April 1992. (Also appears as UCB/ Computer Science Department 92/678)

[Doyle90] J. Doyle, "Rationality and its Roles in Reasoning (extended version)," *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, 1990, pp. 1093-1100.

[Etzioni91] O. Etzioni, "STATIC a Problem–Space Compiler for PRODIGY," *Proceedings of the National Conference on Artificial Intelligence*, Anaheim, CA, July 1991, pp. 533-540,

[Good71] I. J. Good, "The Probabilistic Explication of information, Evidence, Surprise, Causality, Explanation, and Utility,"in *Foundations of Statistical Inference*, V. P. Godambe, D. A. Sprott (cd.), Hold, Rienhart and Winston, Toronto, 1971, pp. 108-127.

[Gratch91] J. Gratch and G. DeJong, "A Hybrid Approach to Guaranteed Effective Control Strategies," *Proceedings of the Eighth International Workshop on Machine Learning,* Evanston, IL, June 1991.

[Gratch92] J. Gratch and G. DeJong, "COMPOSER: A Probabilistic Solution to the Utility Problem in Speed–up Learning," *Proceedings of the National Conference on Artificial Intelligence*, San Jose, CA, July 1992, pp. 235–240.

[Gratch93a] J. Gratch, S. Chien and G. DeJong, "Learning Search Control Knowledge for Deep Space Network Scheduling," *Proceedings of the Ninth International Conference on Machine Learning,* Amherst, MA, June 1993.

[Gratch93b] J. Gratch and G. DeJong, Rational Learning: Finding a Balance Between Utility and Efficiency,, January 1993.

[Greiner92] R. Greiner and I. Jurisica, "A Statistical Approach to Solving the EBL Utility Problem," *Proceedings of the National Conference on A rtificial Intelligence,* San Jose, CA, July 1992, pp. 241–248.

[Horvitz89] E. J. Horvitz, G. F. Cooper and D. E. Heckerman, "Reflection and Action Under Scarce Resources: Theoretical Principles and Empirical Study," *Proceedings of the Eleventh international Joint Conference on Artificial Intelligence,* Detroit, MI, August 1989, pp. 1121-1127.

[Howard70] R. A. Howard, "Decision Analysis: Perspectives on Inference, Decision, and Experimentation," *Proceedings of the Institute of Electrical and Electronics Engineers 58,5 (1970),* pp. 823-834.

[Minton88] S. Minton,in *Learning Search Control Knowledge: An Explanation–Based Approach*, Kluwer Academic Publishers, Norwell, MA, 1988.

[Nadas69] A. Nadas, "An extension of a theorem of Chow and Robbins on sequential confidence intervals for the mean," *The Annals of Mathematical Statistics 40, 2 (1969), pp. 667-671.*

[Provost92] F. J. Provost, "Policies for the Selection of Bias in Inductive Machine Learning," Ph.D. Thesis, Computer Science Department, University of Pittsburgh, Pittsburgh, 1992. (Also appears as Report 92–34)

[Russell89] S. Russell and E. Wefald, "Principles of Metareasoning," *Proceedings of the First International*

Conference on Principles of Knowledge Representation andReasoning, Toronto, Ontario, Canada, May 1989, pp. 400–411.

[Simon76] H.A. Simon, "nom Substantive to Procedural Rat ion alit y,"in *Method and Appraisal in Economits, S.* J. Latsis (cd.), Cambridge University Press, 1976, pp. 129-148.

[Subramanian92] D. Subramanian and S. Hunter,"Measuring Utility and the Design of Provably Good EBL Algorithms," *Proceedings of the Ninth International Conference on Machine Learning,* Aberdeen, Scotland, July 1992, pp. 426-435.

[Tesauro92] G. Tesauro, "Temporal Difference Learni ng of Backgammon Strategy," *Proceedings of the Ninth International Conference on Machine Learning,* Aberdeen, Scotland, July 1992, pp. 451-457.

[Wefald89] E. H. Wefald and S. J. Russell, "Adaptive Learning of Decision-Theoretic Search Control Knowledge," *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, NY, June 1989, pp. 408411.

Appendix

```
Function R4SL ($PE_o$, $R_o$, $\delta$)
[1] i := O; Learn := True; T := Transforms($PE_i$); $\delta^* := \delta/(2|T|)$;
[2] WHILE T # 0 AND Learning-worthwhile (T, $PE_i$) DO                    /* should learning continue? */
[3] { continue := True; j := no; good-rules :=@;
[4]     WHILE T # 0 AND continue DO                                       /* find a good transformation */
[5]     { j:= j + 1; Get $\lambda_j(T,PE_i)$;$\forall t\in$ T: Get $\Delta r_j(t|PE)$;
[6]        significant := Nádas–stopping–rule (T, $PE_i$, $\delta^*$, j);
[7]        T:= T – significant;
[8]        WHEN $\exists t \in$ significant: $\overline{\Delta r_j}(t|PE_i) > 0$          /* beneficial transformation found */
[9]        ( good-rules:= good-rules $\bigcup \{t \in \text{significant} \overline{\Delta r_j}(t|PE_i) > 0\}$
[10]          WHEN $\max_{t\in T}\{\hat{ENP_j}(R_i, t|PE_i)\} \leq \max_{t\in significant}\{\hat{ENP_j}(R_i, t|PE_i)\}$     /* no better transformation */
[11]             best := t $\in$ significant : $\forall w \in$ significant, $\overline{\Delta r_j}(t|PE_i) > \overline{\Delta r_j}(w|PE_i)$ ;
[12]             $PE_{i+1}$ := Apply(best, $PE_i$); T := Transforms($PE_{i+1}$); $\delta^* := \delta/(2|T|)$; /*adopt it*/
[13]             $R_{i+1} := R_i - j \times \overline{\lambda}_j(T, PE_i)$; i := i + 1; continue := False; ) } }
[14] Solve-problems-with-remaining-resources ($PE_i$)

Function Learning-worthwhile (T, PE)
[1] FOR j=1 TO $n_0$ DO
[2]     Get $\lambda_j$(T, PE); $\forall t\in$ T: Get $r_j(PE)$, $\Delta r_j(t|PE)$;
[3] Return $\max_{t\in \Gamma_i}\{\hat{ENP}_{n_0}(R_i, t|PE_i)\} \leq \frac{R_i}{\overline{r}_{n_0}(PE_i)}$

Function Nádas–stopping–rule (T, PE, $\delta^*$, j)
[1] Return $\left\{ t \in T : \frac{S_j^2(t|PE)}{(\overline{\Delta r_j}(t|PE))^2} < \frac{j}{a}, \right\}$ s.t. $\Phi(a)$ $\int_{-\infty}^{a} (1/\sqrt{2\pi}) exp\{-0.5y^2\}dy = \delta^*$ ;
```

Figure 3: Rational COMPOSER Algorithm